Preprocessing for classification of sparse data: application to trajectory recognition

Aurélien Mayoue, Quentin Barthélemy, Sébastien Onis, Anthony Larue

PREPROCESSING FOR CLASSIFICATION OF SPARSE DATA: APPLICATION TO TRAJECTORY RECOGNITION

A. Mayoue, Q. Barthélemy, S. Onis and A. Larue

CEA, LIST, Gif-sur-Yvette, F-91191, France

ABSTRACT

On one hand, sparse coding, which is widely used in signal processing, consists of representing signals as linear combinations of few elementary patterns selected from a dedicated dictionary. The output is a sparse vector containing few coding coefficients and is called sparse code. On the other hand, Multilayer Perceptron (MLP) is a neural network classification method that learns non linear borders between classes using labeled data examples. The MLP input data are vectors, usually normalized and preprocessed to minimize the inter-class correlation. This article acts as a link between sparse coding and MLP by converting sparse code into convenient vectors for MLP input. This original association assures in this way the classification of any sparse signals. Experimental results obtained by the whole process on trajectories data and comparisons to other methods show that this approach is efficient for signals classification.

Index Terms— Sparse coding, classification, multilayer perceptron, trajectories data

1. INTRODUCTION

In signal processing, sparse coding [1, 2, 3] consists of representing signals as linear combinations of few elementary patterns selected from a dedicated dictionary. Firstly, a dictionary building algorithm empirically learns the characteristic patterns associated to a representative database of a signals set. Then, sparse approximation allows to decompose any signals of the considered set with a low number of learned patterns from the dictionary. The sparse representation results in spike series which could be visualized in the form of a spikegram (see Section 3.2). In literature, sparsity is usually considered to make emerge from data elements containing relevant information for compression, denoising and other processings but it lacks for appropriate discrimination tools in a machine learning point of view.

The well-known method Multilayer Perceptrons (MLP) can approximate any decision functions and therefore any forms of border with a simple and effective learning method [4]. This classification method is thus efficient for many supervised learning pattern recognition process. However, classification methods such as MLP usually make use of features in the form of standard vectors (*i.e.* normalized and reduced sized) for analysis. So, it appears that these classical methods are not suited to classify sparse code in the form of spike series.

This paper presents a preprocessing step to transform sparse code into classifier input. This preprocessing is an original projection based method which converts any sparse code from spikegram form into convenient vectors adapted to the MLP input. The proposed approach acts as a link between sparse coding and classical machine learning methods and thus assures the classification of any sparse signals.

In Section 2, we introduce the *Character Trajectories* database that is used throughout the paper to illustrate the different steps of our recognition method. In this way, in Section 3, we explain sparse coding principle which extracts useful primitives from data whereas Section 4 details the preprocessing, prior to classification, which converts sparse code into MLP input vector. Finally, in Section 5 the complete recognition system is applied to the database and performances are compared to those obtained by other classification methods.

2. CHARACTER TRAJECTORIES DATABASE

Experimentations have been done on the free available database *UCI Character Trajectories* (UCI-CT) [5]. It is composed of 2858 pen tip trajectories captured using a WACOM tablet. Three components were kept : v_x and v_y the cartesian velocities, and the pen tip force p'. Only characters without pen up were considered, that is why only 20 characters classes were used instead of 26. One example of each letter is depicted on (Fig : 1). It is noted that velocity signals have been integrated to plot the associated trajectories and that pressure p' is not represented on the figure. In the following, the database has been split in two subsets. Half of the database was used for training whereas the remaining part was kept for the validation step.

Fig. 1. Examples of trajectories from the UCI-CT database.

3. SPARSE CODING METHOD

In this section, we briefly explain the principle of multivariate sparse coding and we present the visualization of sparse code via a spikegram.

3.1. Multivariate Sparse Coding

A signal $y \in \mathbb{R}^N$ of N temporal samples and a normed dictionary $\Phi \in \mathbb{R}^{N \times M}$ composed of M elementary patterns $\{\phi_m\}_{m=1}^M$ are considered. The sparse decomposition of y is done on the dictionary Φ such that :

$$y = \Phi\, x + \epsilon = \sum_{m=1}^{M} x_m \phi_m + \epsilon \quad \text{s.t. } min_x \left\|x\right\|_0 , \tag{1}$$

with $x \in \mathbb{R}^M$ the vector containing the coding coefficients and $\epsilon \in \mathbb{R}^N$ the residual error. As explained in [3], the sparse approximation can be successfully done by pursuit algorithms which select the K strongest energy patterns present in the signal and compute their associated coding coefficients. Thus, the sparse approximation provides a decomposition with only $K (<< M)$ active coefficients, indexed by k :

$$y = \sum_{k=1}^{K} x_{m_k} \phi_{m_k} + \epsilon \, . \tag{2}$$

Dictionary learning [1, 2, 3] is a process which selects and then learns the strong energy patterns appearing in all signals from the studied database. This learned dictionary Φ is thus adapted to the signals

978-1-4673-0183-1/12/$31.00 ©2012 IEEE

set we want to code. Dictionary learning is called sparse coding because, at the end of the learning process, each signal from the database can be sparsely coded on the learned dictionary.

From now, we focus on the particular case of shift-invariance [1, 3]. In this case, patterns are shorter than the signal and are potentially shiftable at all temporal positions of $y(t)$, with t the temporal index for each sample. Thus, the compact dictionary Ψ is only composed of L kernels $\{\psi_l(t)\}_{l=1}^{L}$ which can be translated at all positions τ. Notice that a kernel characterized by a kernel index l and a temporal position τ is named atom : $\psi_l(t-\tau)$. Each signal $y(t)$ of the studied dataset is approximated as a weighted sum of K atoms :

$$y(t) = \sum_{k=1}^{K} x_{l_k,\tau_k}\ \psi_{l_k}(t-\tau_k)\ +\ \epsilon(t)\ , \qquad (3)$$

with x_{l_k,τ_k} the coding coefficients and $\epsilon(t)$ the residue.

Moreover, through our application, studied signals are multivariate, *i.e.* they are composed of several components acquired simultaneously (v_x, v_y and p'). Multivariate dictionary learning [3] is able to deal with multivariate dataset, in such a way that $\mathbf{y}$, ψ_l and ϵ are multivariate signals in the following. A multivariate dictionary Ψ learned from the training set of the UCI-CT database is then displayed on (Fig : 2). Each multivariate kernel is composed of the two cartesian velocities v_x (in dashed line) and v_y (in dotted line) and of the pressure p' (in solid line). At the end of the learning process, the dictionary contains the characteristic multivariate patterns of the database.

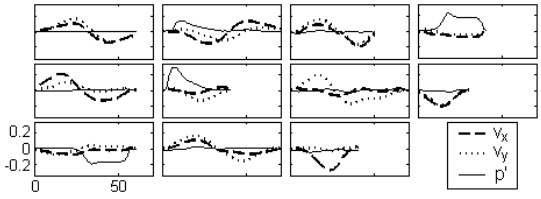

Fig. 2. The $L = 11$ kernels learned on the UCI-CT database. Each kernel is composed of three components (v_x, v_y and p').

3.2. Representation with a spikegram

The sparse decomposition of each *Character Trajectories* signal on the learned dictionary given in (Fig : 2) provides coding coefficients $x_{l,\tau}$. This sparse vector, named sparse code, is usually displayed by a time-kernel representation called spikegram. It condenses three pieces of information : the kernel temporal position τ (abscissa) and index l (ordinate), and the coefficient amplitude $x_{l,\tau}$ (spike color).

Four occurrences of sparse decomposition of the letter q are presented on (Fig : 3). Original (Fig : 3a) and reconstructed (*i.e.* approximated) (Fig : 3b) multivariate signals are displayed. The sparse code is presented in spikegram form on (Fig : 3c). Spikegram can be viewed as the result of the deconvolution of the signal on the learned dictionary.

By comparing the approximated signals to the original ones, we can observe that only few atoms are enough to approximate signals well. The few associated coefficients form the sparse code. Now, we focus on largest amplitude coefficients, (*i.e.* spikes from kernels n° 2, 7 and 9 (with hot color)) and their repetitions show the reproducibility of the decompositions, due to the adapted learned dictionary. Thus, sparsity and reproducibility prove that the representation can be used with interest for a discrimination purpose.

However, the spikegram form is not suitable for standard classification methods such as MLP which usually need normalized vector as input data. The purpose of this paper is to propose a data representation adapted to the Multilayer Perceptron classifier. Notice that the introduced representation can be applied on all spike trains data and not only on those provided by sparse coding methods.

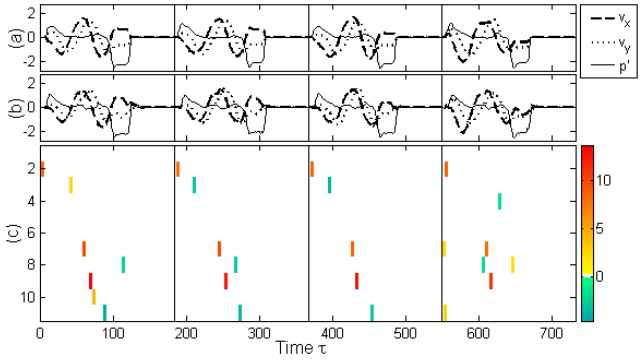

Fig. 3. Original (a) and approximated (b) signals for 4 occurrences of the letter q and the associated spikegram (c).

4. SPARSE DATA CLASSIFICATION

In this section, we briefly remind the properties of the Multilayer Perceptron and then, we detail the preprocessing, prior to classification, which converts sparse code into MLP input vectors to guarantee the use of sparsity for a discrimination purpose.

4.1. Multilayer Perceptron

The Multilayer Perceptron is a flexible machine learning method [4]. It is the most commonly used form of neural networks for classification purpose. Such a network consists of a minimum of three layers : an input layer containing the vector to classify, one or more hidden layers and an output layer whose result vector allows to classify input vector.

Used as a classifier, MLP is able to learn non linear borders between classes of a labeled database. The MLP input data are vectors, usually normalized and often preprocessed to minimize the interclass correlation. In this way, the spikegram representation, which is a $L \times N$ matrix containing the values of the decomposition coefficients $x_{l,\tau}$, is not adapted to be straight classified by MLP.

A first preprocessing method used to fit spikegram to the classification system would be to simply transform the spikegram matrix into a $LN \times 1$ vector by concatenating each line of the matrix. But, the resulting vectors would be too huge to be reasonably processed by classification methods such as MLP. Considering the small size of the database (2858 signals set whose only the half is considered for the training step), the classical methods seem indeed not to be adapted to deal with all temporal shifts which could appear between the spikes series of the same class. That is why the preprocessing method that we present in the following reduces the size of the spikegram while keeping the maximum information.

4.2. Preprocessing prior to classification

Our preprocessing is illustrated on (Fig : 4). In this section, we explain it step by step. Firstly, the sparse code of one occurrence of the letter a, put in the spikegram form, is considered on (Fig : 4a). Then, we define a temporal Hanning window of fixed size T as :

$$\text{Hanning}(t) = \begin{cases} \frac{1}{2}\left(1 + cos\left(\frac{2\pi}{T}\ t\right)\right), & t \in \left[-\frac{T}{2}, \frac{T}{2}\right] \\ 0 & \text{otherwise} \end{cases} . \qquad (4)$$

The spikegram is paved using J temporal Hanning windows as shown on (Fig : 4b). The center of the first window corresponds to the appearance of the first atom $\psi_{l_1}(t-\tau_1)$ and the following windows are successively shifted by the half of the window size (*i.e.* with an overlapping of $T/2$). Zeros are added to each processed signal so that the number of windows J stays the same whatever the initial length of the

signal. After that, all signals have the length N_{max} of the longest signal of the database. Parameter J is empirically chosen (see Section 5.2) and $T = \frac{N_{max}}{0.5\ J}$ where 0.5 means an overlapping of 50% between the Hanning windows.

Next, we produce the reduced matrix M_r of size $L \times J$ displayed on (Fig : 4c), with $j = 1 \,..\, J$ the window index. The element $M_r(l, j)$ corresponds to an atom projection of kernel index l on the Hanning window centered at $\tau_1 + (j-1) \times \frac{T}{2}$ (where τ_1 is the first atom position). So, $M_r(l, j)$ is defined by (Eq : 5) where τ_k is the position of the k^{th} atom and x_{l_k,τ_k} its associated coefficient :

$$M_r(l,j) = \sum_{k=1}^{K} \text{Hanning}\left(\tau_k - \tau_1 - (j-1)\frac{T}{2}\right) x_{l_k,\tau_k} \text{ s.t. } l_k = l. \quad (5)$$

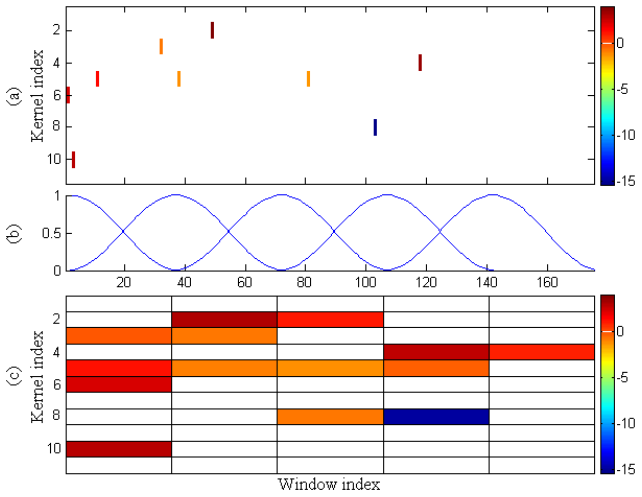

Fig. 4. Spikegram data preprocessing. (a) is a spikegram obtained for one trial of the letter a, (b) represents the Hanning windows and (c) the matrix $M_r(l, j)$ obtained by the projection of the spikegram on the Hanning windows.

This representation presents several advantages. The data size is small and fixed without significant loss of information. Indeed, according to an hypothesis of sparsity, we can assume that, in practice, there is only one atom from the same kernel ψ_l in each Hanning window. Consequently, the exact atom position τ_k and the associated coefficient x_{l_k,τ_k} can be recovered from the values $M_r(l, j)$ and $M_r(l, j+1)$. It is noted that this last remark is only true for an overlapping of 50% because the sum between the Hanning windows always equals 1 in this particular case and so :

$$x_{l_k,\tau_k} = M_r\,(l,j) + M_r\,(l,j+1)\ , \quad (6)$$
$$\text{s.t. } l_k = l \text{ and } \tau_k \in \left[\tau_1 + (j-1)\frac{T}{2}, \tau_1 + j\frac{T}{2}\right]\ .$$

Finally, each matrix M_r is horizontally vectorized into vector v and normalized such that :

$$v((l-1)J + j) = M_r(l,j)\ , \text{ and } v = \frac{v}{\|v\|_2}\ . \quad (7)$$

The reduced vector v is the result of our preprocessing which transforms spikegram representation without loss of information. In its final form, it can now be used as input of a standard machine learning system. Some examples of vector v are illustrated on (Fig : 5). One occurrence of each letter is displayed on (Fig : 5a) whereas 40 occurences of the same letter a show the repetition of the vectors v for the same class (Fig : 5b).

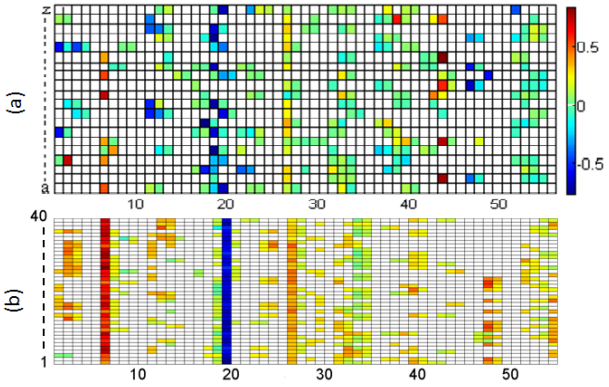

Fig. 5. Reduced vectors v computed on sparse code from the UCI-CT database. (a) Each row corresponds to the vector v representative of one letter (from a to z). (b) Each row corresponds to the vector v representative of one (among 40) occurence of the letter a.

5. EXPERIMENTATION

In this section, experimental context for classification is first presented. Then, we explain how parameters are chosen. Finally, our approach is evaluated and compared to other classification methods which have been already applied to the *UCI Character Trajectories* database.

5.1. Experimental context

Experimentations have been done on the UCI-CT database. It is reminded that 50% of the data were randomly used to train the dictionary Ψ and the MLP (Training Set TS), while the other part was kept for the test phase (Validation Set VS), repeating the randomly selection ten times to increase the results precision. Data are divided in 20 classes corresponding to the 20 letters. The class index is denoted by c.

In order to compare performance of the MLP classification method on sparse data, we apply two other classification systems to the extracted vectors v : Mean Vectors and Perceptron methods. Both systems are linear projections, *i.e.* each vector v is projected onto 20 reference vectors V_c for each of the 20 letters from the database. The class with the highest projection value is then the given system classification result : $\hat{c} = \arg\max_c(V_c \cdot v)$, where $V_c \cdot v$ is the inner product between the c class reference vector V_c and the vector v to be classified.

The first projection method named 'Sparse Coding + Mean Vector' is a simple generative system which consists of describing each class c as the mean of all training set vectors of the class c.

$$V_c = \text{mean}(v),\ v \in TS \text{ s.t. } \text{class}(v) = c\,. \quad (8)$$

The second projection method named 'Sparse Coding + Perceptron' consists of finding the reference vectors V_c which best discriminates the different classes c to each other. The vectors V_c are computed using a Perceptron. This method is a two layers MLP (a MLP without hidden layer and with linear borders). The Perceptron has 20 output neurons, *i.e.* one neuron for each class c. The activation function is $\varphi = \tanh(.)$. The Perceptron is trained (update of V_c and bias b_c) on TS so that the value z_c of the c^{th} output neuron equals 1 if the class of the input vector v is c while other output neurons equal -1. Thus, the Perceptron minimizes the quadratic error E between the Perceptron output z_c and the target value t_c for a vector v of class c according to :

$$z_c(v) = \tanh(V_c \cdot v + b_c)\ , \quad (9)$$
$$E = \sum_{v \in TS} (z_c(v) - t_c)^2\ , \quad (10)$$
$$\left[\hat{V_c}, \hat{b_c}\right] = \arg\min_{[V_c,b_c]}(E)\ . \quad (11)$$

Finally, we evaluate the MLP classification system, with 20, 50 and 100 hidden neurons. The higher the number of hidden neurons is, the more MLP can learn complex borders. However with the low number of database examples, the overfitting risk increases. According to Perceptron, MLP has 20 output neurons (one for each letter). The MLP target output t_c and input v are the same as for the Perceptron.

5.2. Parameters choice

We explain the choice of our parameters and see their influence on the MLP recognition rate. We set the same spikegrams and feature preprocessing parameters for all experimentations. As explained in [3], parameters for sparse coding are set in order to obtain a good compromise between square error of signals approximation and the sparsity characterized by the number of kernel L and the number of active atoms K. Following experiments on *TS*, we choose $L = 11$ kernels for the dictionary and a maximum of $K = 10$ atoms for decompositions.

The only preprocessing parameter is the number of Hanning windows J. It is set to maximize the recognition rate of the complete system (see Table 1). In this way, we set $J = 5$ windows for each signal, what gives $L \times J = 55$ elements for each vector v. Moreover, as mentioned in Section 4.2, an overlapping of 50% between the Hanning windows is the best choice to have no significant loss of information during the preprocessing step. We validate this choice through experiments carried out on the UCI-CT database with different values for the overlapping (see Table 2).

Table 1. Recognition rates obtained by a MLP (20 hidden neurons) on UCI-CT database for different numbers J of Hanning windows (with an overlapping of 50%).

Nb of windows	3	4	5	6	7
Recognition rate (%)	94.5	95.2	96.3	96.2	96.1

Table 2. Recognition rates obtained by a MLP (20 hidden neurons) on UCI-CT database for different values of overlapping between the Hanning windows (with $J = 5$).

Overlapping (%)	30	40	50	60	70
Recognition rate (%)	95.9	96.1	96.3	95.9	95.7

5.3. Evaluation

On top of the classification methods (Mean Vector, simple Perceptron and MLP) presented in Section 5.1 and applied on the sparse code, three state-of-the-art classification methods have been evaluated on this database (but without sparse coding) : a Top Kernel (TK) [6], a Fisher Kernel (FK) [7] and a Hidden Markov Models (HMM) based method [8]. Recognition rates obtained for all trajectories classification systems on the Validation Set of the UCI-CT database are resumed in Table 3. We can see that the association of Sparse Coding and a discriminative method (Perceptron or MLP) outperforms other systems with respectively 95.3% and 97.1% of good classification. Moreover, we obtain better results than the state-of-the art (TK, FK and HMM) with linear classification system (Perceptron) and a recognition rate near to 80% with a simple 'Mean Vector' projection method. Thus, sparse coding preserves discriminative information and preprocessing provides convenient vectors for classification.

Concerning the projection classification methods, the 'Perceptron' and the 'Mean Vectors' approaches are quite different. The 'Mean Vectors' reference vectors V_c are learned to best describe each class. On the contrary, the 'Perceptron' learning phase provides reference vectors V_c which best discriminate the different classes to each other. The results prove that information used to describe the trajectories are less relevant than those used to perform discrimination in a classification purpose.

Regarding the number of hidden neurons for the MLP, we note that the recognition rates are not very sensitive to this parameter. Indeed, multiplying by 5 the number of hidden neurons improves the results of only 0.8%. A number of 100 hidden neurons seems a good compromise between the MLP complexity and the recognition rate.

Table 3. Recognition rates on UCI-CT database for the state-of-the-art methods (TK, FK and HMM) and different settings of our method.

Methods	Recognition rate
Sparse Coding (SC) + Mean Vector	78.3 %
SC + Perceptron	95.3 %
SC + MLP (20 Hidden Neurons)	96.3 %
SC + MLP (100 Hidden Neurons)	97.1 %
SC + MLP (200 Hidden Neurons)	96.7 %
TK [6]	93.67 %
FK [7]	89.26 %
HMM [8]	92.91 %

6. CONCLUSION

In this paper, we have presented an original trajectories classification system based on a preprocessing method that is necessary to adapt sparse code into convenient normalized vector for classification. The sparse coding approach is explained, showing its relevance to describe multivariate signals. Then, we have described an original preprocessing which keeps the sparse coding advantages while providing easily exploitable inputs for standard classification methods. Finally, this system has been applied to trajectories recognition and has been compared to other classification systems. The results illustrate the efficiency of preprocessed sparse coded data for multivariate signals classification tasks, especially used as Perceptron or MLP input. With our original preprocessing approach, this paper can be considered as a first step to link classification methods to sparse coding.

7. REFERENCES

[1] E. Smith and M.S. Lewicki, "Efficient auditory coding," *Nature*, vol. 439, pp. 978–982, 2006.

[2] A. Aharon, M. Elad, and A. Bruckstein, "K-SVD : an algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[3] Q. Barthélemy, A. Larue, A. Mayoue, D. Mercier, and J I. Mars, "Shift & 2D rotation invariant sparse coding for multivariate signals," *IEEE Trans. on Signal Processing*, vol. 60, pp. 1597–1611, 2012.

[4] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, Wiley-Interscience, second edition, 2001.

[5] A. Frank and A. Asuncion, "UCI machine learning repository," http ://archive.ics.uci.edu/ml, 2010.

[6] T S. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Proc. Conf. on Advances in Neural Information Processing Systems II*, 1999.

[7] K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K R. Müller, "A new discriminative kernel from probabilistic models," *Neural Comput.*, vol. 14, pp. 2397–2414, 2002.

[8] A. Perina, M. Cristani, U. Castellani, and V. Murino, "A new generative feature set based on entropy distance for discriminative classification," in *Proc. of the 15th International Conference on Image Analysis and Processing ICIAP '09*, 2009.